SPRING BEAN SCOPES

When defining a <bean> in Spring, you have the option of declaring a scope for that bean. For example, To force Spring to produce a new bean instance each time one is needed, you should declare the bean's scope attribute to be **prototype**. Similar way if you want Spring to return the same bean instance each time one is needed, you should declare the bean's scope attribute to be **singleton**.

The Spring Framework supports following five scopes, three of which are available only if you use a web-aware ApplicationContext.

Scope	Description
singleton	This scopes the bean definition to a single instance per Spring IoC container (default).
prototype	This scopes a single bean definition to have any number of object instances.
request	This scopes a bean definition to an HTTP request. Only valid in the context of a web-aware Spring ApplicationContext.
session	This scopes a bean definition to an HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.
global-session	This scopes a bean definition to a global HTTP session. Only valid in the context of a web-aware Spring ApplicationContext.

This chapter will discuss about first two scopes and remaining three will be discussed when we will discuss about web-aware Spring ApplicationContext.

The singleton scope:

If scope is set to singleton, the Spring IoC container creates exactly one instance of the object defined by that bean definition. This single instance is stored in a cache of such singleton beans, and all subsequent requests and references for that named bean return the cached object.

The default scope is always singleton however, when you need one and only one instance of a bean, you can set the **scope** property to **singleton** in the bean configuration file, as shown below:

```
<!-- A bean definition with singleton scope -->
<bean  >
    <!-- collaborators and configuration for this bean go here -->
</bean>
```

Example:

Let us have working Eclipse IDE in place and follow the following steps to create a Spring application:

Step	Description
1	Create a project with a name *SpringExample* and create a package *com.tutorialspoint* under the **src** folder in

	the created project.
2	Add required Spring libraries using *Add External JARs* option as explained in the *Spring Hello World Example* chapter.
3	Create Java classes *HelloWorld* and *MainApp* under the *com.tutorialspoint* package.
4	Create Beans configuration file *Beans.xml* under the **src** folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **HelloWorld.java** file:

```java
package com.tutorialspoint;

public class HelloWorld {
   private String message;

   public void setMessage(String message){
      this.message  = message;
   }

   public void getMessage(){
      System.out.println("Your Message : " + message);
   }
}
```

Following is the content of the **MainApp.java** file:

```java
package com.tutorialspoint;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context = 
             new ClassPathXmlApplicationContext("Beans.xml");

      HelloWorld objA = (HelloWorld) context.getBean("helloWorld");

      objA.setMessage("I'm object A");
      objA.getMessage();

      HelloWorld objB = (HelloWorld) context.getBean("helloWorld");
      objB.getMessage();
   }
}
```

Following is the configuration file **Beans.xml** required for singleton scope:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <bean
      scope="singleton">
   </bean>

</beans>
```

Once you are done with creating source and bean configuration files, let us run the application. If everything is fine with your application, this will print the following message:

```
Your Message : I'm object A
Your Message : I'm object A
```

The prototype scope:

If scope is set to prototype, the Spring IoC container creates new bean instance of the object every time a request for that specific bean is made. As a rule, use the prototype scope for all state-full beans and the singleton scope for stateless beans.

To define a prototype scope, you can set the **scope** property to **prototype** in the bean configuration file, as shown below:

```
<!-- A bean definition with singleton scope -->
<bean  >
    <!-- collaborators and configuration for this bean go here -->
</bean>
```

Example:

Let us have working Eclipse IDE in place and follow the following steps to create a Spring application:

Step	Description
1	Create a project with a name *SpringExample* and create a package *com.tutorialspoint* under the **src** folder in the created project.
2	Add required Spring libraries using *Add External JARs* option as explained in the *Spring Hello World Example* chapter.
3	Create Java classes *HelloWorld* and *MainApp* under the *com.tutorialspoint* package.
4	Create Beans configuration file *Beans.xml* under the **src** folder.
5	The final step is to create the content of all the Java files and Bean Configuration file and run the application as explained below.

Here is the content of **HelloWorld.java** file:

```java
package com.tutorialspoint;

public class HelloWorld {
   private String message;

   public void setMessage(String message){
      this.message  = message;
   }

   public void getMessage(){
      System.out.println("Your Message : " + message);
   }
}
```

Following is the content of the **MainApp.java** file:

```
package com.tutorialspoint;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
   public static void main(String[] args) {
      ApplicationContext context =
             new ClassPathXmlApplicationContext("Beans.xml");

      HelloWorld objA = (HelloWorld) context.getBean("helloWorld");

      objA.setMessage("I'm object A");
      objA.getMessage();

      HelloWorld objB = (HelloWorld) context.getBean("helloWorld");
      objB.getMessage();
   }
}
```

Following is the configuration file **Beans.xml** required for prototype scope:

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <bean
      scope="prototype">
   </bean>

</beans>
```

Once you are done with creating source and bean configuration files, let us run the application. If everything is fine with your application, this will print the following message:

```
Your Message : I'm object A
Your Message : null
```